Complexité structurelle – TD 12
Probabilistically checkable proofs

On rappelle ci-après la définition des classes PCP. $L \in \mathrm{PCP}(r(n), q(n))$ s'il existe un vérificateur V, utilisant $r(n)$ bits aléatoires et lisant $q(n)$ bits de la preuve de manière non-adaptative, tel que :

- si $x \in L$ alors il existe une preuve Π tel que $V^{\Pi}(x)$ accepte toujours ;
- si $x \notin L$ alors pour toute preuve Π, $\mathrm{Proba}(V^{\Pi}(x) \text{ accepte}) \leq 1/2$.

On rappelle enfin le théorème PCP :

$$\mathrm{NP} = \mathrm{PCP}(O(\log n), O(1)).$$

Exercice 1

Max-k-CSP est le problème suivant : étant données les tables de vérité de t fonctions booléennes $f_1, \ldots, f_t$ d'arité k sur n variables, retourner une affectation qui maximise le nombre de f_i satisfaites. Montrer que Max-k-CSP est NP-complet dès que $k \geq 3$. En admettant le théorème PCP, montrer qu'il existe k tel que pour tout $\epsilon > 0$, si l'on peut $(2-\epsilon)$-approximer Max-k-CSP, alors P = NP.

Soit QPSAT le langage constitué de l'ensemble des uples de polynômes $(f_1, \ldots, f_n)$, où les $f_i \in \mathbf{F}_2[X_1, \ldots, X_n]$ sont de degré ≤ 2, tels qu'il existe une racine commune :

$$\exists a \in (\mathbf{F}_2)^n \text{ tel que } \bigwedge_{i=1}^{n} (f_i(a) = 0).$$

Exercice 2 Le but de cet exercice est de montrer que $\mathrm{NP} = \mathrm{PCP}(n^{O(1)}, O(1))$.

1. Montrer que QPSAT est NP-complet. Dans la suite, on va montrer que QPSAT $\in \mathrm{PCP}(O(n^2), O(1))$.
2. Pourquoi l'idée naïve de donner comme preuve la racine commune ne fonctionne pas ?
3. Pour $r, a \in (\mathbf{F}_2)^n$ on pose $Q(r,a) = \sum_i r_i f_i(a)$. Avec quelle probabilité sur r a-t-on $Q(r,a) = 0$?
4. Puisque le degré de $Q(r,.)$ est ≤ 2, à r fixé on peut écrire
$$Q(r,a) = \alpha + ({}^t\beta)a + ({}^ta)\gamma a,$$
où α est un scalaire de $\mathbf{F}_2$, β un vecteur et γ une matrice. On fournit la preuve $\Pi = (\Pi_1, \Pi_2)$, avec
$$\Pi_1 = ({}^t\beta a)_{\beta \in (\mathbf{F}_2)^n} \text{ et } \Pi_2 = ({}^ta\gamma a)_{\gamma \in (\mathbf{F}_2)^{n \times n}}.$$
Proposer un algorithme pour le vérificateur. Pourquoi ça ne marche toujours pas ?
5. Il faut que $\beta \mapsto \Pi_1(\beta)$ soit linéaire, ainsi que $\gamma \mapsto \Pi_2(\gamma)$. Supposons que ce soit le cas. Il faut par ailleurs que Π_1 et Π_2 soient définis à partir du même a. Proposer un test pour cela. Pour prouver la correction de ce test, on pourra étudier la probabilité sur r, r' que $({}^tr)Ar' = ({}^tr)Br'$ pour deux matrices A et B différentes.
6. Proposer un test probabiliste de linéarité : il s'agit de vérifier qu'une fonction f donnée par un oracle est linéaire.
7. Résumer le protocole. Note : il faut en plus utiliser une « auto-correction » des fonctions Π_1 et Π_2. Pour évaluer une fonction f (prétendue linéaire) sur x, l'autocorrection consiste à retourner la valeur $f(x+y) - f(y)$ avec un y choisi aléatoirement.